Performance Characterization of an Automated System for Robot Configuration Synthesis

Chris Leger[a]
Science and Technology Development Section, Jet Propulsion Laboratory
California Institute of Technology, Pasadena CA

ABSTRACT

We describe the results of several experiments aimed at understanding the performance, behavior, and limitations of Darwin2K, an automated system for robot configuration synthesis and optimization. Two design tasks are addressed: the design of a robot for walking along trusses in zero gravity, and of a manipulator for a coverage task. We explore the impact of several factors on synthesizer performance, including the set of robot components and assemblies available to the synthesizer, the set of metrics used to quantify robot performance, and the scope of the task on which robots are assessed. The meaning and impact of the experimental results is given, and we discuss potential improvements in the method of use of Darwin2K as well as future improvements to the system itself.

Keywords: Automated design, evolutionary algorithms, design optimization, robot design

1. INTRODUCTION

We have developed an automated design tool, called Darwin2K, for robot configuration synthesis and optimization. Darwin2K's development was driven by the desire for an effective and widely applicable software system for automatically performing robot configuration design, a system that could produce useful results for a range of applications. Briefly, the system uses an evolutionary algorithm to create and optimize robot designs, including kinematic, dynamic, structural, actuator, and other properties. The evolutionary algorithm modifies and combines known configurations using *genetic operators*, and decides which configurations should be modified or combined on the basis of robot performance so that "good" robots are selected in preference to "bad" ones. Robot performance is assessed by task-specific simulation with respect to multiple metrics, which are selected according to a task's needs. Darwin2K represents robots as assemblies of *parameterized modules*, allowing robot of varying complexity (including single, multiple, or bifurcated serial chains), type (mobile robots, manipulations, and combinations thereof), and construction (modular or monolithic) to be created. The software architecture is also modular, allowing task-specific simulations to be quickly constructed, and is extensible so that new software components and robot modules can be easily added. Finally, Darwin2K is implemented in a distributed manner, allowing dozens of computers to be brought to bear on difficult design problems. (The flip side is that given current desktop computer power, a dozen or two computers are usually *required* to reduce system runtime to hours rather than days!) Detailed descriptions of the internals of Darwin2K are given elsewhere[1]; our goal in this paper is to give the reader an understanding of the use, applicability, capabilities, and limitations of Darwin2K. To this end, we will present the results of a series of synthesis trials for two design tasks.

1.1 Related work

The use of evolutionary algorithms for robot design and optimization has been the focus of a number of other research efforts. Earlier systems focused on one of two areas: modular design or kinematic design. Modular design systems[2-6] constructed manipulators or mobile robots from a set of modules with fixed properties, while kinematic design systems[7-10] only represented the kinematic properties of robots (i.e. Denavit-Hartenberg parameters or equivalent) and did not account for dynamics, actuators, or structural properties. Both approaches have their limitations: design with fixed modules necessarily results in design compromises as module properties are predetermined and thus cannot be changed to improve performance, while purely kinematic methodologies are limited in design scope (i.e. no non-kinematic properties are determined by the system) and in the usefulness of their results, since subsequent "fleshing out" of designs to include actuators, link structure, and dynamics is dependent on kinematic design and may require modification of the kinematic configuration to achieve a feasible result. More recently, the simultaneous synthesis of robot structure and control programs by an evolutionary algorithm has been investigated, though the intent of this effort has

a. The author was at the Robotics Institutes at Carnegie Mellon University when this research was performed. This work was funded by a National Science Foundation Graduate Research Fellowship.

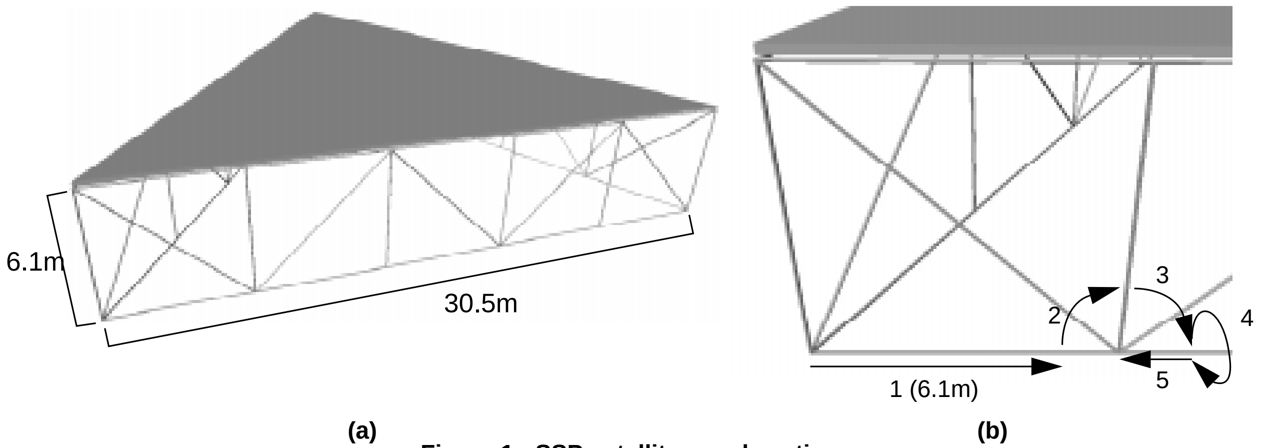


Figure 1: SSP satellite panel section

(a) The space solar power satellite's transmitter array is composed of a number of panels, each in the shape of an equilateral triangle and backed by a truss to provide rigidity. (b) During the inspection task, the robot will walk along one of the truss segments, transition to several other segments, and inspect the junction between them.

been the creation of small robots capable of self-propelled motion rather than design of machines for optimally performing a particular task or function[11]. A similar approach has also been taken in the context of computer graphics for the creation and animation of "virtual creatures" which interact with each other in simulation, resulting in surprising, efficient, and entertaining creatures and behaviors[12]. Darwin2K builds upon and differs from these other approaches by combining modular and kinematic design to allow more detailed and flexible synthesis of modular and non-modular robots of varying topology. Additionally, Darwin2K includes more complete and detailed robot descriptions and simulation capabilities, such as forward dynamic simulation (computation of robot motion based on applied actuator forces and robot inertial properties) and estimation of link deflection due to actuator, inertial, and applied forces and torques. Finally, Darwin2K investigates new methods for optimization of multiple performance metrics and constraints. As mentioned in the introduction, these features have been described elsewhere; this paper focuses instead on characterizing and explaining Darwin2K's performance on synthesis tasks.

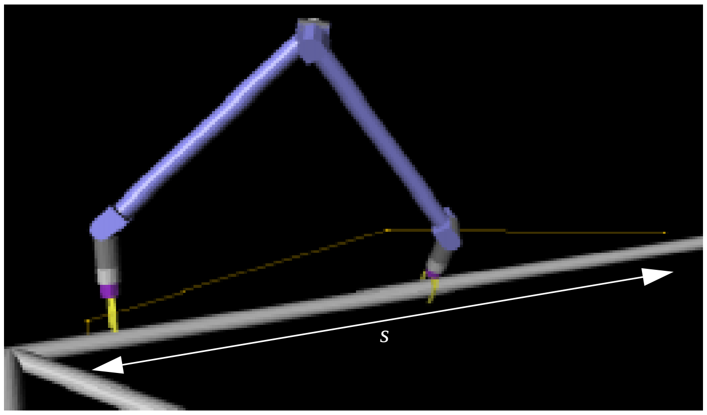
2. TASK 1: A WALKING ROBOT FOR SPACE TRUSSES

Recently, there has been a renewed interest in Space Solar Power (SSP), orbiting solar power satellites that beam energy to the Earth in the form of microwaves. The geosynchronous orbits in which the satellites would lie are far beyond the Space Shuttle's range; thus, automated assembly and servicing are therefore important enabling technologies for SSP. By current thinking, each power satellite will have a modular construction and will be quite large, on the order of kilometers. Much of the satellite's structure will be composed of trusses, upon which the antenna array and solar collectors will be mounted. According to one model, the antenna array will be assembled from triangular sections each consisting of a panel backed by a truss to provide rigidity (Figure 1a).

The first task we will address is the synthesis of a lightweight walker for visual inspection of the satellite's components. Robotic walkers are preferred over free-flyers for operations on the power satellites, as walkers do not rely on expendable fuel. An inspection robot should be able to move to any point on the truss: it should be able to walk along truss rods, transition between them, and change its orientation with respect to a truss rod so that it can move from one side of the truss to the other. The robot should also be able to position its end effectors so that it can aim a camera at the object being inspected (we will assume there is a camera mounted on each effector, though the cameras will not be modeled in the simulation).

The representative task for the inspection robot is shown schematically in Figure 1b and encompasses the capabilities listed above. The robot first walks along a truss segment 6.1m in length, moves to a perpendicular segment (gripping the segment 1.0m from the truss joint), moves to a third segment, rotates around to the back side of the truss, and then moves one end effector to an inspection location under the truss joint. This task ensures that the robot can reasonably maneuver about the truss in addition to being able to walk along truss segments.

During synthesis, each candidate robot is simulated as it performs the task. A Jacobian-based controller (specifically, one that uses the Singularity Robust Inverse[13,14]) computes the joint velocities and accelerations required to make the robot's grippers follow a Cartesian gait trajectory, and then uses a dynamic model of the robot to compute



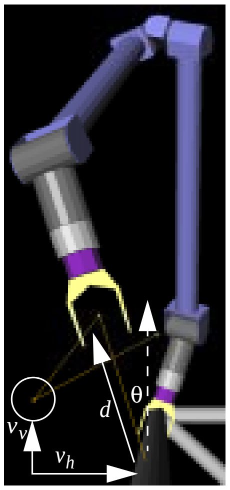


Figure 2: Gait parameters

These two images illustrate the parameters for each stride of the walker's gait:

- s - distance between successive grasp points for an end-effector (stride)
- d - distance from segment center to approach point
- θ - angle from normal to approach point
- v_v - offset of via point (white circle) in direction of the normal (the dotted line in the right image)
- v_h - offset of via point perpendicular to the normal
- v_{tol} (not shown here) is an additional gait parameter for how close the end effector must come to the via point.

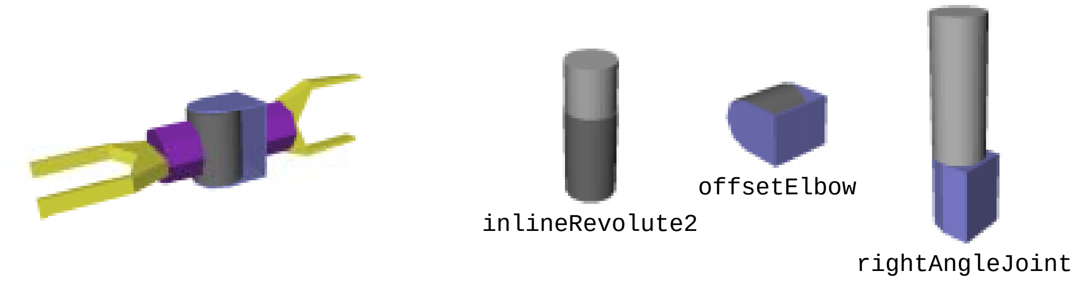
Metric name	Priority	Criteria
`dynamicPathCompletion`	0	= 1.0 (100%)
`collisionMetric`	0	integral = 0
`linkDeflectionMetric`	1	max < 0.001m
`continuousSaturationMetric`	1	max < 0.8 (80%)

Metric name	Priority	Criteria
`massMetric`	2	none
`powerMetric`	2	none
`timeMetric`	2	none

Table 1: Metrics for SSP inspection robot

the requisite torque commands[1]. The gait trajectory is parameterized so that the synthesis algorithm can optimize it; Figure 2 depicts the gait and its geometric parameters. The velocity and acceleration to use when following the gait trajectory are also included as parameters to be optimized by Darwin2K. A fully dynamic simulation is used to compute the robot's motion, so that the effects of actuator saturation (finite actuator torque), robot dynamic properties, and controller behavior can be guaged.We will ignore the mechanics of grasping, as a special-purpose gripper can be used and its impact on the rest of the robot will be minimal.

Darwin2K groups metrics by priority, and focuses on optimizing more important metrics before bothering with less important metrics. For example, minimizing robot mass or energy use is pointless if the robot cannot complete the task at all, and so any robot that can complete the task should be considered better than any robot that cannot. For the current task, seven metrics are used to quantify robot performance, as shown in Table 1. A lower priority number in the table indicates greater importance. The first four metrics (those with a priority number of 0 or 1) are considered to be constraints: they have acceptance criteria that must be met in order for a robot to be deemed *feasible*. The first of these is the `dynamicPathCompletionMetric`, which measures the fraction of the task that is completed by a robot. The `collisionMetric` measures the integral over the simulation of the number of collisions between the robots links and the truss, or with other links; clearly this metric should be zero if the robot is to be considered feasible. After satisfying these two metrics, the synthesizer focuses on the `continuousSaturationMetric` (which ensures that the robot's actuators are operating within their acceptable continuous torque limits) and `linkDeflectionMetric` (which measures the deflection of the robot's end effectors due to link bending, allowing link cross-sections to be adequately sized). Finally, the last three metrics (mass, energy, and time) are not constraints but rather are quantities that should be minimized, and are considered only after a sufficient number of feasible candidates have been generated.



(a) kernel configuration (b) modules

Figure 3: Kernel configuration and modules for the SSP inspection robot
(a) shows the kernel (initial) configuration from which all other robots are derived, through insertion of the modules shown in (b) and subsequent exchange of modules between robots. The three modules each have one degree of freedom and associated motor and gearhead parameters (which select components corresponding to off-the-shelf products from Maxon and HD Systems), and additionally have geometric parameters (e.g. overall length) so that no link modules are required.

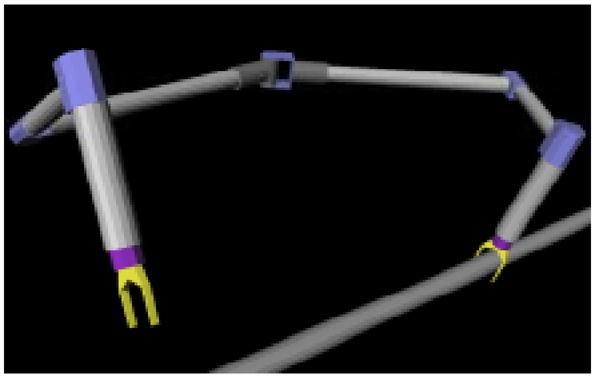
So far, we have looked at only half of the problem specification: the task description (in the form of the simulation details) and performance metrics. The other half of the specification consists of the set of robot modules used to construct robots, and constraints on how the modules are connected. First, we will require robots to have two symmetric limbs, as this is the minimum number of limbs to provide continuous contact with the truss during locomotion. The *kernel configuration* (shown in Figure 3a) is the initial robot from which all other robots are derived, and specifies a joint structure to which the two symmetric limbs should be attached, with each limb ending in a gripper. The space of possible configurations is also influenced by the set of allowable modules (Figure 3b). Each of the modules contains parameters for geometric dimensions and motor and gearhead properties, and the modules are inserted into and exchanged between robot configurations (starting with the kernel) in order to produce novel robots.

The robot must have at least six degrees of freedom between its two end effectors so that it can grasp truss segments and position the camera in arbitrary orientations and in three dimensions; however, it does not need six degrees of freedom in each limb as there is no need for body positioning in this task. We will include an extra degree of freedom to improve manipulability, and will thus require 7 DOF in the robot mechanism. This is accomplished with a *configuration filters* which eliminates any configuration that does not have exactly 7 degrees of freedom. Another filter disallows successive joints whose axes are colinear, as such joints do not contribute to the robot's manipulability.

The first phase of synthesis was for Darwin2K to randomly generate 5000 robots, measure their performance, and then discard all but the best 200, as determined by the first two metrics. For the rest of the experiment, a population of 200 robots was used: that is, at any point in time Darwin2K kept a list of 200 robots (and their performance measurements) from which new robots were created. We used idle CPU time from approximately 20 workstations (each with roughly 200-500 MIPS of computing power), and set a time limit of 13 hours for the synthesis run. The system generated the first configuration satisfying the initial set of metrics (task completion and collision-free motion) after evaluating only 100 robots beyond the original 5000 configurations. To prevent the system from converging too rapidly on a solution, we required a minimum of 4000 robots to be created and evaluated between the time the first feasible robot (with respect to a group of metrics) is created, and the time the next group of metrics is addressed. By the time 4000 more robots had been created, 152 out of 200 robots in the population satisfied the first two metrics and the second set of metrics were activated. The first configurations satisfying the second set of metrics were generated after 1,500 further evaluations (for a total of 10,600 evaluations), and after another 4000 the synthesizer advanced to the third set of metrics with a total of 70 feasible configurations.

At this point, the lightest feasible configuration had a mass of 22.9kg; the most energy-efficient configuration consumed 955J to complete the task, and the fastest configuration completed the task in 85 seconds. The synthesis process continued and the time limit was reached after a total of 52,500 evaluations, including the 5000 for the initial population. At this point the lightest, most energy-efficient, and fastest configurations had significantly better performance: the lightest configuration had a mass of 12.2kg, the most efficient used 221J, and the fastest required only 55 seconds to complete the task (Figure 4). Figure 5 shows the fastest robot at several stages while executing the task.

The robot in Figure 4 seems to have an odd configuration: the three degrees of freedom for each "wrist" are distributed along the length of the arm, rather than being closely-grouped at the end of the arm. The first module of each arm is an `inlineRevolute2`, and the last two are `rightAngleJoints`. 197 of the robots in the final population shared the same configuration graph topology; the remaining three were similar to the majority except that an



***cfg$_t$* - lowest completion time**
mass - **18.6** kg
energy - **1112** J
time - **55.2** s
key differences:
- high velocity and acceleration parameters
- powerful wrist actuators enable higher accelerations
- larger-diameter links provide stiffness for high accelerations

***cfg$_m$* - lowest mass**
mass - **12.2** kg
energy - **255** J
time - **77.2** s
key differences:
- small-diameter links are lightweight;
- small, lightweight actuators

***cfg$_e$* - lowest energy**
mass - **12.3** kg
energy - **221** J
time - **84.7** s
key differences:
- lower velocity and acceleration parameters than *cfg$_m$* (and *cfg$_t$*) lead to lower energy use

Figure 4: Walkers with lowest mass, energy, and task completion time
The feasible configuration with lowest task completion time is shown at upper left, with its performance and salient features listed at upper right. The performance and key differences of the lightest and most efficient robots are given below. All three configurations share the same kinematic structure (7 DOFs total) that inherently minimizes self-collisions; they differ only in parameter values, not topology.

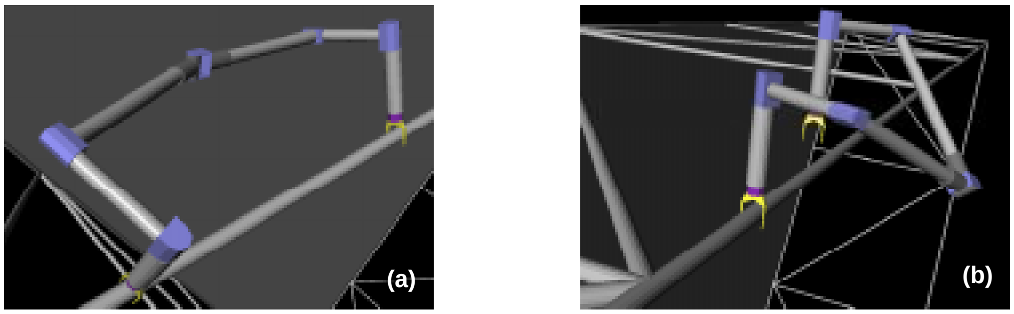


Figure 5: Synthesized walker executing inspection task
(a) robot walking along truss and (b) preparing to grip the third segment with both grippers before rotating

`inlineRevolute2` module was oriented differently, resulting in slightly different inertial properties. The main drawback of this configuration is that changing the orientation of the end-effector in three dimensions requires large motions, since each of the wrist joints (and corresponding links) must be moved; however, such motions are required only infrequently in this task. Are there any benefits to this configuration? After examining the kinematics of the arm, it becomes apparent that this configuration is excellent for avoiding self-collisions: when considering each half of the arm separately, each of the three wrist joints can be moved through a full revolution without causing any self-collisions (Figure 6). This is not generally the case for 3-DOF wrists with multiple intersecting axes, as are commonly found in dextrous manipulators. The robot's inherent lack of self-collisions due to wrist motion make the control problem easier, which is particularly important given that the controller used in the experiment does not try to avoid self-collisions. There has been recent work in identifying and exploiting "mechanical intelligence" to perform roles normally reserved for active control, specifically for dynamic stability in mechanisms[15,16]; the kinematic configuration evolved in this experiment may be said to possess a similar type of mechanical intelligence for avoiding self collisions. This kinematic configuration is also an example of generating novel, unintuitive designs: it seems likely that a human designer would not give high priority to inherent collision avoidance the design process, and would instead plan to use an appropriate controller to perform collision avoidance.

To see if these results were repeatable, or if any other unanticipated configurations could be synthesized, we

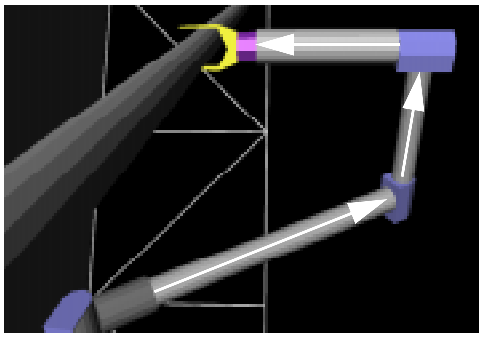


Figure 6: Close-up of walker wrist
The wrist structure present in the evolved walkers allows full rotation about each of the three wrist axes without any self-collisions. The joint axes are indicated by the arrows.

	mass	energy	time
lightest	13.7 kg	482 J	69.4 s
most efficient	15.6 kg	263 J	87.3 s
fastest	15.3 kg	917 J	54.9 s

Table 2: Performance of best walkers from second trial

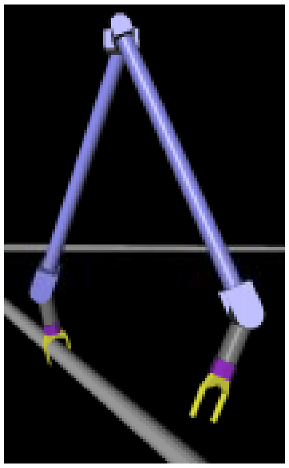


Figure 7: Best robots evolved for walking-only task
The robots that were evolved only for walking (without the transition or inspection motions) do not have the collision-minimizing wrist assembly synthesized in earlier runs. The fastest robot is shown at left, and performance is also given for the most energy-efficient and lightest robots.

Metric	Task	Mass	Energy	Time
Mass	complete	12.2 kg	98 J	30.5 s
	walking only	6.3 kg	78 J	40.5 s
Energy	complete	12.3 kg	82 J	33.7 s
	walking only	12.3 kg	38 J	46.3 s
Time	complete	18.6 kg	468 J	19.8 s
	walking only	14 kg	479 J	19.6 s

Table 3: Comparison of robots for walking-only task
The robots evolved for the walking-only task performed significantly better in one metric (shaded table cells) when compared to those evolved for the full inspection task.

conducted a second trial, resulting in robots having the same collision-minimizing kinematic configuration as those from he first trial, though the actual modules used are different: whereas in the first experiment each limb consisted of an `inlineRevolute2` followed by two `rightAngleJoints`, the configurations in the second experiment reversed the order of the arm modules with the two `rightAngleJoints` coming first followed by the `inlineRevolute2`. The performance measurements for the best feasible configurations in the second experiment are shown in Table 2. As in the first trial, the synthesizer ran for 13 hours; however only 35,000 evaluations, rather than 52,000, were performed (due to fewer available computers for evaluation) and thus slightly lower robot performance in terms of mass and energy is not surprising. Note that the fastest configuration (at 54.9s) was slightly faster than in the first trial (55.2s) and is both lighter and more energy efficient.

It seems likely that a more efficient walker can be synthesized if the only requirement is to be able to walk along truss segments and not transition to the back side of the truss. We can expect the robots synthesized only for walking to be better at walking than the robots we have seen so far, but perhaps be unable to perform transitions. To test this hypothesis, the synthesizer was re-run with a modified task specification that did not include any transitions between segments. The robots synthesized for this task (Figure 7) have a markedly different structure than those evolved for the full inspection task. The three joints at the intersection of the two limbs are similar to human hips, though they have one less degree of freedom and cannot independently lift a leg without changing its orientation, and

Metric	Metrics used	Mass	Energy	Time
Lowest mass	all	12.18 kg	255 J	77.2 s
	mass only	14.4 kg	569 J	73.6 s
Lowest energy	all	12.3 kg	221 J	84.7 s
	energy only	19.6 kg	537 J	87.6 s
Lowest time	all	18.6 kg	1112 J	55.2 s
	time only	24.5 kg	1506 J	60.5 s

Table 4: Comparison of walkers optimized for single vs. multiple metrics
Starting from the same set of feasible configurations, the synthesizer produced better-performing robots when simultaneously optimizing mass, energy, and time, than when optimizing each metric independently.

the wrists each have two intersecting joint axes. As detailed in Table 3, the lightest, most energy-efficient, and fastest robots have significantly better performance in at least one metric each than their counterparts evolved for the full inspection task. (The robots that were evolved for the full inspection task were re-evaluated on the walking-only task to produce the performance measurements in the table.) When the robots evolved only for walking are evaluated on the full task, they all have self-collision problems at the wrist when rotating about the truss segment. This demonstrates a property of task-specific synthesis that can be both a strength and a weakness: by definition, the robots are specialized for the task and, while they may perform quite well at the desired task, slightly different tasks may present substantial problems for the robots. For this reason, it is crucial to ensure that the representative task used during the evaluation process is in fact representative of the operations that will be encountered in the actual application.

In the experiments above, we have examined the impact of the task description on synthesis results. Another factor that influences the results of the synthesis process is the set of metrics used to measure performance. In the previous experiments there were multiple competing metrics (mass, time, and energy) in the final set of metrics to be optimized. Do these work against each other, resulting in configurations that are a compromise between the different metrics? Or do they have a beneficial effect on robot performance by pulling the synthesizer in different directions so that local minima in a single metric can be avoided? To answer these questions, we conducted three more synthesizer trials, once for each metric in the final set of metrics (mass, energy, and time). Each trial began with the population from the first experiment at the point the final three metrics were added; however, instead of three metrics, the final set of metrics contained only a single metric in each of the three trials (mass, energy, or time).

Even though the single-metric experiments were run for 40,000 evaluations, the robots with lowest mass, energy, and time performed significantly worse than the best from the original, three-metric experiment (Table 4). It thus appears that multiple metrics may improve the synthesizer's performance rather than degrading it. One potential reason for this surprising result is that multiple metrics lead the synthesizer in different directions and result in a more diverse population of configurations, thus improving the synthesizer's ability to escaping local optima with respect to any single metric. While we have not experimentally verified this hypothesis, it seems to be the most likely explanation for the substantial and repeated performance differences between the single- and multiple-metric experiments.

3. TASK 2: A FIXED-BASE MANIPULATOR

One example of automated task-based kinematic synthesis from Kim[8] is the design of a manipulator for waterproofing the tiles on the underside of the Space Shuttle. For this task, the underside of the Space Shuttle is demarcated into a number of rectangular regions that will be serviced sequentially by a manipulator mounted on a mobile base. Each region measures 3m across its diagonal and contains approximately 180 tiles, each of which must be individually waterproofed by the robot's manipulator after the mobile base is repositioned beneath the region. Using Darwin2K to synthesize manipulators for this task will allow us to compare its results to those of Kim's system (which also only addressed synthesis of the robot's manipulator), and the moderate complexity of this task makes it useful for examining the impact of different starting conditions such as modules and kernel configuration on synthesis results.

As with Kim's work, we will simulate the robot over a single representative region that contains the range of heights and orientations that will be encountered on the underside of the Shuttle. The representative region is 2.14m on a side, and ranges from a horizontal surface 3m above the ground to a 45 degree surface 4m above the ground (Figure 8) The number of tiles reached by the manipulator during simulation is also reduced from 180 to 7 (in Kim's work) and 16 (in this experiment) tiles which span the representative region so that the entire necessary workspace is covered.

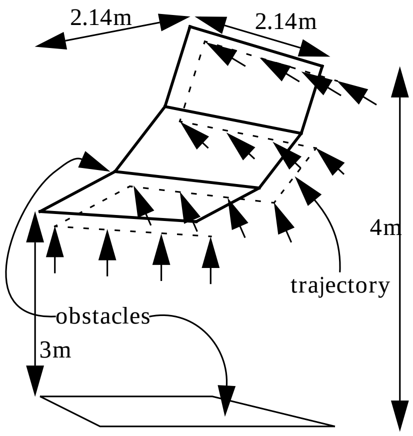


Figure 8: Trajectory and obstacles for Space Shuttle waterproofing manipulator
The polygons represent the workspace obstacles, while the trajectory is shown as a dotted line with arrows indicating the tool orientation at each via point.

Metric name	Priority	Acceptance threshold
`pathCompletionMetric`	0	= 1.0 (100%)
`collisionMetric`	0	integral = 0
`positionErrorMetric`	0	max < 0.03m
`linkDeflectionMetric`	1	max < 0.002m
`actuatorSaturationMetric`	1	max < 1.0 (100%)
`massMetric`	2	none
`timeMetric`	2	none

Table 5: Metrics for Space Shuttle waterproofing manipulator

	1-DOF revolute joints	`prismatic-Tube`	`scaraElbow`	# kernels	kernels (see Figure 9)
Baseline	●			1	(a)
Prismatic	●	●		1	(a)
SCARA	●	●	●	1	(a)
Const	●			1	(d)
High level	●	●	●	3	(a), (b), and (c)

Table 6: Modules and kernels for manipulator experiments

The waterproofing mechanism at the robot's end effector is modeled as a 1.5kg payload, and to ensure accurate placement of the payload a position error tolerance of 2mm is used at each via point in the trajectory -- that is, the end effector must stop within 2mm of each via point before moving to the next one.

The metrics used for this synthesis problem are listed in Table 5. The first requirement group is composed of the `pathCompletionMetric`, `collisionMetric`, and `positionErrorMetric`, which together ensure that the entire trajectory is accurately followed with no collisions. The second requirement group contains the `linkDeflectionMetric` and `actuatorSaturationMetric` so that link structure and joint actuators may be appropriately sized, and finally the third requirement group consists of the `massMetric` and `timeMetric` so that an efficient, lightweight robot can be created. Since the manipulator's base is not free-flying and we are requiring actuator saturation to be less than one (i.e. each joint torque command must be less than the torque rating of the corresponding actuator), we can use kinematic rather than dynamic simulation for this task.

To investigate the impact of module and kernel choice on the synthesizer's performance, five different sets of experiments (with five runs each) were performed: Baseline, Prismatic, SCARA, Const, and High-Level. The same task and simulator were used for all five sets of experiments; the differences were in the set of allowable modules, and in the topologies supplied in the kernel configuration. A unique initial population was generated for each run of the synthesizer by using different seed values to initialize the random number generator. Table 6 summarizes the different module and kernel properties for each of the experiments, while Figure 9 depicts the kernel configurations used in each of the experiments. The Baseline, Prismatic, and SCARA experiments all used a single simple kernel consisting of a `simpleTool` mounted directly on a `fixedBase`. The Const experiment used a kernel duplicating the constraints used in Kim's experiment: the first joint axis was vertical, and the last three (wrist) axes intersected in a point. The High-Level experiment used the same simple kernel as in the Baseline, Prismatic, and SCARA experiments and also included two other kernels: one had two `elbowJoints` forming a shoulder, and the other had a wrist assembly identical to that in the Const experiment. The `dofFilter` was used to limit the number of degrees of freedom for the

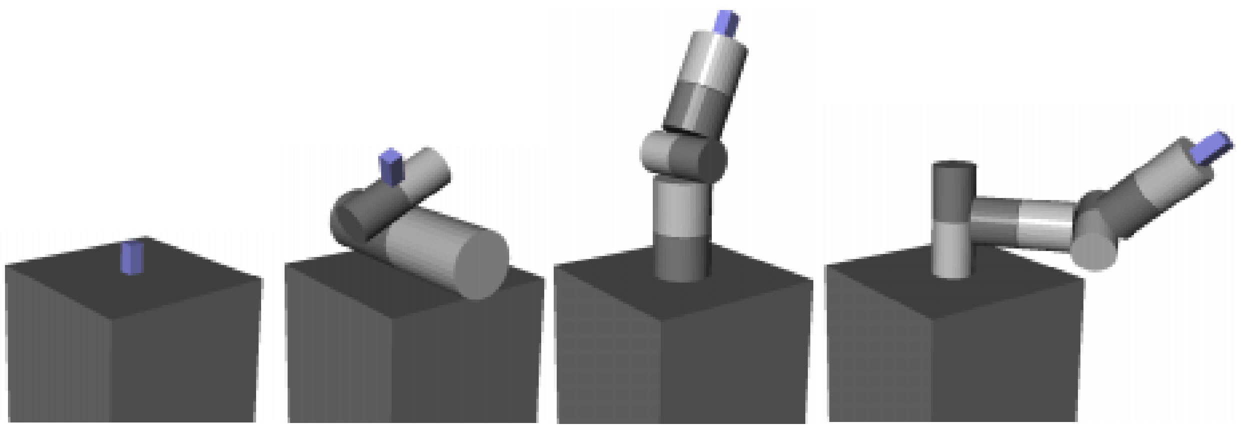


Figure 9: Kernels used in the manipulator experiments
(a) shows the simple kernel used in all experiments except for Const, which used (d) instead. The High-Level experiment also included one kernel with a shoulder assembly (b) and another with a wrist assembly (c).

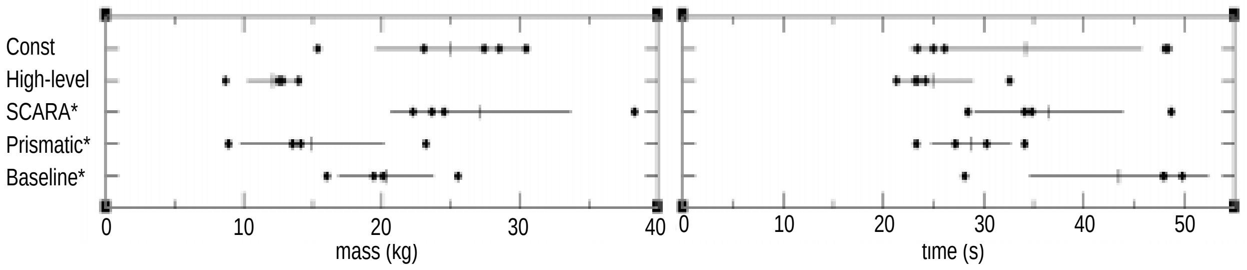


Figure 10: Mass and time for varying starting conditions
In the upper graph, each dot indicates the mass of the lightest configuration generated for each successful run of the five different sets of experiments. The vertical line shows the mean mass over the successful experiments, and the horizontal line indicates +/- one standard deviation. The lower graph is similar, but for task completion time. The experiments with an asterisk next to their label (SCARA, Prismatic, and Baseline) only had four successful runs out of five; one of each of their runs failed to produce any feasible configurations in the time allowed.

Const experiment to 7 (as was used in Kim's experiment), and between 5 and 7 for the other experiments. The terminating conditions for each experiment were a maximum run time of 6 hours and a minimum of 80,000 configurations; between 80,000 (for most of the High-Level experiments) and 140,000 configurations (for the unsuccessful runs in other experiments) were evaluated during each run. As with the walker experiment, parameters for velocity and acceleration were included with each robot so that they could be optimized by the system.

A summary of the best time and mass of any feasible configuration synthesized during each five-run experiment is shown in Figure 10. The High-Level experiments consistently produced the best results -- the average of the best mass and time over the five trials is significantly lower than for the other experiments, and the standard deviations of mass and time are lower than for the other experiments indicating more consistent performance. A rough ordering of the other experiments in terms of the average of best mass and time over the successful trials is Prismatic, Const, Baseline, and finally SCARA. As we will see, these results can be understood by considering the how the primitives available for each experiment influence the synthesizer's ability to find feasible configurations, and how well-optimized the first feasible configurations are.

Figure 11 shows typical robots from the final populations generated from the various initial conditions. The topology of the High-Level and Prismatic results are generally similar: the basic form is a 2-DOF shoulder followed by a prismatic joint and ending in a 2- or 3-DOF wrist. In terms of mass, the solutions generated in the High-Level experiments are superior to all other experiments, with the Prismatic results close behind. Based on the similarity of the topologies from the Prismatic and High-Level experiments, one can surmise that the use of a prismatic joint is beneficial for creating lightweight manipulators for this task. The results for task completion time are similar though not as one-sided, primarily because task completion time is heavily influenced by the task parameters for maximum ac-

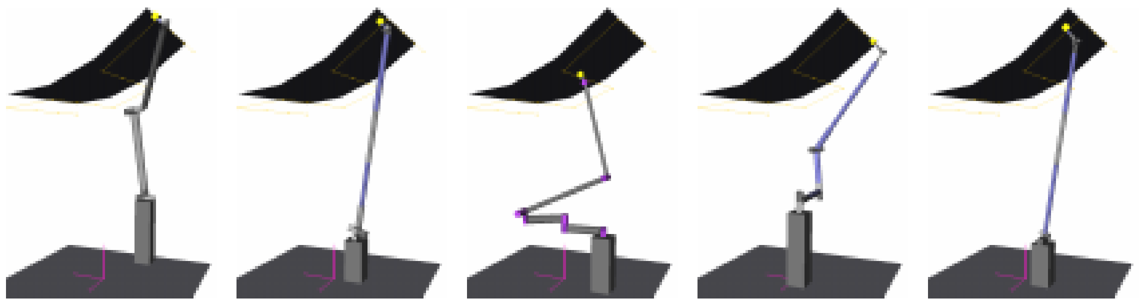


Figure 11: Typical results from different starting conditions
Shown here are representative robots from five trials with different starting conditions. The High-Level and Prismatic robots were, on average, significantly lighter than others, and slightly faster.

celeration and velocity along the path. Task completion time is thus less dependent on topology (and therefore on available modules) than on task parameters, though a configuration's actuators must be able to supply the joint velocities necessary to follow the trajectory at the speed dictated by the task parameters if maximum performance is to be achieved.

The lack of a prismatic joint can be seen as the major reason that the Const and Baseline experiments were not able to produce configurations as light as those in the High-Level and Prismatic experiments. However, the SCARA experiments also included a `prismaticTube` in the module database and yet the best configurations generated in the SCARA experiments were substantially inferior to those in the High-Level and Prismatic experiments. None of the feasible SCARA robots in the final population contain a prismatic joint, which is at first surprising given that the High-Level and Prismatic experiments made effective use of the `prismaticTube` module. The SCARA experiments did make heavy use of the `scaraElbow` module, though: all of the feasible configurations in four of the runs contained two `scaraElbow` modules, and in the remaining experiment all feasible configurations had one `scaraElbow` module. In the SCARA experiments, the synthesizer was able to quickly generate configurations which satisfied the first set of requirements. The synthesizer quickly focused on these configurations and became trapped in a local minimum. The high complexity of the SCARA module (three joints and two links) makes it easy for the synthesizer to create feasible configurations, and many feasible configurations simply consist of the fixed base, two `scaraElbow` modules, and the tool. When feasible configurations can be created very easily from complex modules, they can quickly dominate the selection of configurations for reproduction. Any major changes to these configurations -- such as those necessary to create configurations using a prismatic joint as in the High-Level experiments -- will likely result in poorly-performing configurations that have little chance of reproduction and will quickly be culled from the population. It is thus easy for the synthesizer to become trapped in a local minimum when promising designs can be quickly synthesized from complex modules that are not actually useful for well-optimized configurations. On the other hand, when well-suited for a task complex modules with many parameters will likely require less synthesis time compared to simpler modules with few parameters. The designer should thus be willing to use complex modules if there is good reason to believe they will be useful for a particular task.

Complex topological features can likewise improve synthesis speed and quality when appropriate for the task. An examination of the best configurations from the High-Level experiments reveals that the assemblies specified in the kernel configurations were highly useful: the fastest and lightest configurations in every High-Level trial contain both the wrist and shoulder assemblies from the kernels. The High-Level experiments were able to quickly generate configurations satisfying the first two sets of requirements, substantially faster than the Prismatic experiments though the resulting robots from the two experiments were similar. The reduced synthesis time and improved performance of the resulting configurations clearly indicate the utility of supplying human knowledge to the synthesis process in the form of useful subassemblies in the kernel configurations.

The Const experiments lie somewhere between the High-Level and SCARA experiments, in terms of both the quality and the phenomena responsible for the results. The Const experiments contained complex primitives (in this case, the kernel configurations) which, like the `scaraElbow` module, turn out to be detrimental to performance. The Const experiments were constrained to use only revolute joints, have 7 degrees of freedom, and use the vertically-oriented shoulder axis and 3-DOF wrist specified in the kernel configuration. These constraints prevent solutions similar to those in the High-Level and Prismatic experiments from being discovered, though they do seem to be somewhat helpful in that the task completion times of the best configurations are better for the Const experiments than for the Baseline experiments even though both used the same set of modules. The number of evaluations required to generate configurations satisfying the first and second requirement groups is also lower for the Const experiment than for the Baseline experiment, indicating that the wrist and shoulder subassemblies in the Const kernel reduced the amount of exploration required by the synthesizer when generating feasible configurations. Similarly, the High-Level experiments required less search than the Prismatic experiments, even though they arrived at similar results.

4. DISCUSSION

4.1 Summary

In this paper, we applied Darwin2K to two design tasks: a Space Shuttle waterproofing manipulator and a truss walker. Through these tasks, we explored the sensitivity of the synthesizer to changes in the module database, kernel configuration, task, and evaluation metrics. The walker experiments demonstrated synthesis of unexpected yet well-optimized and potentially novel configurations, showing the utility of automated synthesis for tasks that are not well-understood on the basis of previous experience within the design team.

On the other hand, human expertise still plays an important role in determining the quality of synthesis results. Including useful topologies and modules can greatly improve the performance of the synthesized robots, while including sub-optimal modules or topological features can lead to poor performance. Finally, the synthesizer generates a wide range of feasible configurations with different trade-offs between multiple metrics, and it is the designer's job to choose among these based on the requirements of the task.

Some areas for improvement are obvious. The synthesis process would benefit from the use of optimal, or at least deliberative, planning and control, as local controllers inevitably introduce biases into the synthesis process by not making best use of each robot. Improved dynamic simulation that included closed kinematic chains and simple terrain interaction models would extend Darwin2K's utility for mobile robot synthesis problems. Additionally, the author's recent experience in assisting a third party in the use of Darwin2K points to the need for ease-of-use issues to be addressed, such as documentation, portability, and user interface design.

4.2 The role of Darwin2K in the design process

In the experiments presented in this paper, Darwin2K performed configuration design, a significant portion of the design process. However, there are other phases of design that occur both before and after configuration design: conceptual design, and detailed design. Conceptual design decides the general form of the machine (or machines) that will address a task: Will a mobile robot or fixed-base manipulator be used? Will there be one robot, or several? Should the mobile robot walk, roll, or fly? For each of the experiments, conceptual design was implicitly performed in the process of describing the task, kernel configurations, and configuration filters: for example, the truss walker experiments assumed symmetric walkers with 7 degrees of freedom, rather than a free-flying camera or some sort of articulated, wheeled robot.

The effort required to specify tasks and implement task-specific planning or control algorithms is the main reason Darwin2K did not address high-level conceptual design. The addition of flexible planners, or of motion plan synthesis performed simultaneously with configuration synthesis, would allow Darwin2K to perform conceptual design. This would reduce the effort required for task specification, most likely at a substantial increase in computational cost. Even with these proposed improvements, it may not make sense to give Darwin2K an unrestricted (or very loosely-restricted) design space; rather, it may be more useful to apply Darwin2K to each conceptual design sequentially. For example, wheeled, flying, and walking robots for the truss inspection task would have drastically different topological and parametric features, and it is not likely that exchanging information between them would prove useful. Furthermore, simultaneous exploration of all three schemes for mobility would likely result in convergence upon a single modality and would not allow each method to be explored and optimized to the fullest extent. In contrast, using Darwin2K to explore, detail, and optimize each modality independently would provide the designer with a realistic assessment of the performance and trade-offs possible with each design. This approach simultaneously allows a more informed conceptual design decision to be made, and results in a complete configuration description that is ready for detailed design.

Partitioning of the design space is also useful at a finer level: for example, the experiments that explored the impact of module and kernel choice indicated that manipulators with prismatic joints were better-suited for the task than manipulators with only revolute joints. Using Darwin2K in a sequential fashion starting from different configuration alternatives allows each alternative to be explored in-depth and optimized. In contrast, manual design methods often make mid-level configuration decisions (e.g. revolute or prismatic joints, number of degrees of freedom, etc.) before each alternative can be extensively evaluated. When a robot is being designed manually, the mid-level configuration decisions may often be made when each alternative is at the “stick figure” level of detail. Darwin2K provides much more information about the pros and cons of each alternative and provides a significantly more detailed description of the configuration, allowing the designer to make more appropriate mid-level configuration decisions and thus avoid surprises down the road as the chosen configuration is fleshed out. This difference between manual and automated design is partially due to an interesting and fundamental difference in design methodology: manual design starts with vague design concepts and proceeds to refine and detail each concept, discarding alternatives before they are fully specified, while automated approaches such as Darwin2K use the same level of detail for every design and seek to improve performance by changing attributes of the design.

The development, testing, and characterization of Darwin2K focused on a highly-automated manner of use that did not take advantage of human interaction during the synthesis process. While this resulted in a capable synthesis system, it is likely that the computational resources required for synthesis can be substantially reduced by including more human interaction during synthesis. One mode of interaction is to use Darwin2K in an iterative manner: the best few configurations from one run of the synthesizer can be used as the kernel configurations for another run. The designer can make some of the parameters and topological features constant in the new kernel configurations in order to limit the search space, thus allowing Darwin2K to be used in a coarse-to-fine manner. Another mode of interaction would be the “Hand of God”, which would allow the designer to view and modify configurations while the synthesizer is running so that easily-remedied design flaws could be quickly corrected. This would allow the designer to influence the exploration of the design space and would probably be most useful in helping the synthesizer escape local minima.

5. REFERENCES

1. C. Leger, “Darwin2K: An Evolutionary Approach to Automated Design for Robotics”, Kluwer Academic Press, 2000.
2. R. Ambrose and D. Tesar. The optimal selection of robot modules for space manipulators. In *Proceedings of the ASCE Space 94 Conference*, February 1994.
3. I. Chen. On optimal configuration of modular reconfigurable robots. In *Proceedings of the 4th International Conference on Control, Automation, Robotics, and Vision*, 1996.
4. O. Chocron and P. Bidaud. Genetic design of 3d modular manipulators. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 223–228, April 1997.
5. S. Farritor, S. Dubowsky, N. Rutman, and J. Cole. A systems-level modular design approach to field robotics. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 2980–2895, 1996.
6. C. Paredis. *An Agent-Based Approach to the Design of Rapidly Deployable Fault Tolerant Manipulators*. PhD thesis, The Robotics Institute, Carnegie Mellon University, 1996.
7. P. Chedmail and E. Ramstein. Robot mechanism synthesis and genetic algorithms. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 3466–3471, 1996.
8. J.-O. Kim and P. Khosla. Design of space shuttle tile servicing robot: An application of task-based kinematic design. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pp. 867–874, 1993.
9. A. McCrea. Genetic algorithm performance in parametric selection of bridge restoration robot. In *Proceedings of the 14th International Symposium on Automation and Robotics in Construction*, pages 437–441, 1997.
10. C. Paredis and P. Khosla. Kinematic design of serial link manipulators from task specifications. *The International Journal of Robotics Research*, 12(3):274–287, June 1993.
11. H. Lipson and J. B. Pollack. Towards Continuously Reconfigurable Self-Designing Robotics. In *Proceedings of the 2000 IEEE Conference on Robotics and Automation*, 2000.
12. K. Sims. Evolving virtual creatures. In *1994 Computer Graphics Proceedings*, pages 15–22, 1994.
13. L. Kelmar and P. Khosla. Automatic generation of forward and inverse kinematics for a reconfigurable modular manipulator system. *Journal of Robotic Systems*, 7(4):599–619, 1990.
14. Y. Nakamura. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, September 1986.
15. T. McGeer. Passive dynamic walking. *International Journal of Robotics Research*, 9(2):62–82, April 1990.
16. G. Zeglin. *The Bow-Leg Hopping Robot*. PhD thesis, The Robotics Institute, Carnegie Mellon University, 1999.
17. K. Dowling, R. Bennett, M. Blackwell, T. Graham, S. Gatrall, R. O’Toole, and H. Schempf. A mobile robot for ground servicing operations on the space shuttle. In *OE / Technology 1992 Cooperative Intelligent Robots in Space*. SPIE, November 1992.